NATIONAL UNIVERSITY OF SINGAPORE

Semester 2, 2017/2018

CS1010 Programming Methodology

Time Allowed: 2 Hours

INSTRUCTION TO CANDIDATES

1. This is an OPEN book assessment.
2. This assessment paper contains **FIVE (5)** questions and comprises THIRTEEN **(13)** printed pages.
3. Answer *ALL* questions within the spaces provided in this booklet.
4. You are allowed to use the back of the paper but please remember to state "P.T.O."
5. *Cross out any draft* or otherwise we will mark the poorer answers.
6. Please write your student number below, but NOT your name.

We reserved our rights to deduct marks if:

1. Your writing is too untidy or unrecognisable, or
2. Your code is too lengthy. Basically our guideline is, if your code is two times longer than our model answer, it would be considered as too long.

STUDENT NUMBER:____________________

(This portion is for examiner's use only)

Question	Max. Marks	Score	Check
Q1	10		
Q2	8		
Q3	15		
Q4	13		
Q5	14		
Total	60		

Question 1 (10 marks)

Calculate the mode of heights of Singaporeans divided into genders. The "mode" of a set of data is the value that occurs most often. You can assume that there are N Singaporeans in the database and the function `readDataBase()` will read the data into two arrays `gender[]` and `height[]`. For example, if the database contains the data of 100 Singaporeans, height[99] will store the height of the last person in *cm* (in **integer**) and gender[99] will store the character 'M' if he is a male or 'F' otherwise.

Your program should compute and print out the modes for male and female separately. For example, if the data of 10 persons is like:

Indices	0	1	2	3	4	5	6	7	8	9
gender	M	M	F	F	F	M	M	M	M	F
height	170	180	180	170	170	180	190	180	190	190

Your program should output:

```
There are 10 persons in the database.
The mode for female Singaporeans is 170 cm.
The mode for male Singaporeans is 180 cm.
```

You can assume
- All Singaporeans are shorter than 200cm in the database.
- All the heights are given in integers.
- The modes are unique in each gender.
- The database contains both genders, and at least 1000 persons in each gender.

You can declare any extra valuables or functions but you **CANNOT** declare any new array. If you declare any new variables, please put some comments on their purposes.

```
#include <stdio.h>
// N is the number of persons in the database
#define N 57731



int main(void) {

  int height[N];
  char gender[N];
  int F[999]={0}, M[999]={0}; // For your own use
  int modeFemale, modeMale;
  int i,j,k;









  // program continues on the next page
```

Question 1 Cont.

```
  readDataBase(N, height, gender);
  // you can assumed that this function is implemented for you already
  // And this will read the data into the two arrays, height and gender
```

```
  printf("There are %d persons in the database.\n", N);
  printf("The mode for female Singapores is %d cm.\n",modeFemale);
  printf("The mode for male Singapores is %d cm.\n",modeMale);
}
```

Question 2 (8 marks)

Trace the code and write down the expected printout for each box.

```
#include<stdio.h>

void func1(char* s);
void main() {
  char *title = "CS1010";
  func1(title);
  printf("\n");
}

void func1(char* s) {
  if(*s) {
    func1(s+1);
    printf("%c",*s);
  }
}
```

Expected printing:

```
#include <stdio.h>


int main() {
        int list[] = {9,3,2,8,5,6,7,4,1,10};
        int i,temp;

        for (i=1;i<10;i++)
                if (list[i] > list [i-1])
                {
                        temp = list[i];
                        list[i] = list[i+1];
                        list[i+1] = temp;
                }
        for (i=0;i<10;i++)
                printf("%d ",list[i]);
        printf("\n");
}
```

Expected printing:

```
#include <stdio.h>

int main(void) {
        int n = 9;
        if (n > 10)
                if (n > 11)
                        printf("A\n");
        else
                printf("B\n");
}
```

Expected printing:

```
#include<stdio.h>
void mystery(int, int*, int[]);
void main() {
      int a = 1, *ptr_b, arr_c[] = {2, 3, 4};
      ptr_b = arr_c;
      mystery(a, ptr_b, arr_c);
      printf("%d %d %d %d %d\n", a, *ptr_b, arr_c[0], arr_c[1], arr_c[2]);
}

void mystery(int a, int *ptr_b, int arr_c[]){
      ptr_b = &(ary_c[2]);
      ptr_b--;
      *ary_c = *ptr_b;
      ptr_b = &a;
      *ptr_b++;
      a+=2;
      *ptr_b = a;
      return;
}
```

Expected printing:

Question 3 (4+5+6 marks)

Suppose that we are to write a student information system that stores the following information for each student:

Matric number:	The matriculation number of the student, limited to 8 characters.
Name:	The name of the student, limited to 50 characters.
Gender:	The gender of the student, which is either 'M' or 'F'.
GPA:	The GPA of the student, which is a real number in [0, 5].
Department:	The school that the student belongs to, limited to 50 characters.

a) Define a structure **`student_t`** to store information for each student.

```
#define MATRIC_LEN = 8;
#define MAX_STRING_LEN = 50;
```

Question 3 (Cont.)

b) Write a function **`read_file()`** that reads student information from a specified file. A sample of the file's content is shown below:

```
A1234567
Alice Tan
F
4.50
Department of Computer Science
A2345678
Bob Sim
M
4.2
Department of Information Systems
A3456789
Cathy Leong
F
3.7
Department of Computer Science
...
```

The first line is a matric number of a student, followed by the student's name, gender, GPA, and department, each on a new line. Each student's information takes up five lines. The information of subsequent students follow immediately. The function declaration is:

`int read_file(char filename[], student_t stu_list[])`

The first argument is the filename, the second one is the array of students. This function returns the number of students read.

```
int read_file(char filename[], student_t stu_list[]) {
```

Question 3b (Cont.)

Question 3 (cont.)

c) Write a function **`sort_stu()`** to sort an array of students in ascending order of their matric numbers, assuming that all matric numbers are unique. The function header is given below:

```
#include <string.h>

...

void sort_stu(student_t arr_stu[], int stu_num) {
```

Question 4 (7+6 marks)

Given two strings S and T, we may change S to T by removing some characters from S and inserting some new characters into it. We define the *distance* between S and T as the minimum number of character removals/insertions required to change S to T. For example, the distance between "ABCD" and "ADEFC" is 5, since we can change the former to the latter using 5 character removals/insertions as follows:

1. "ABCD" → "ACD", i.e., removing 'B'.
2. "ACD" → "AD", i.e., removing 'C'.
3. "AD" → "ADE", i.e., inserting 'E'.
4. "ADE" → "ADEF", i.e., inserting 'F'.
5. "ADEF" → "ADEFC", i.e., inserting 'C'.

As another example, the distance between "XYZ" and "ZYX" is 4.

We need a **recursive** function that takes two strings as input, and returns the distance between the two strings. No mark will be given if your answer is not based on recursion.

For your information the base case and recurrence relation of the recursion is as follows. Let S and T to denote the two input strings. Let S' denote the string obtained by removing the first characters from S, and T' denote the string obtained by removing the first characters from T. Let $dist(X, Y)$ to denote the distance between two strings X and Y.

Base case: If S is empty or T is empty, then $dist(S, T)$ equals the length of the longer one among S and T

Recurrence relation: If the first characters of S and T are the same, then $dist(S,T) = dist(S',T')$; otherwise, $dist(S, T)$ equals the smallest one among the following three:

1. $1 + dist(S', T)$
2. $1 + dist(S, T')$

a) Explain why the above base case and recurrence relation are correct.

Question 4 (cont.)

b) Write a definition of the recursive function in C using the following skeleton, i.e., `Dist`. The function has two input parameters: pointers to two strings `str1` and `str2`, both of length at most 100. The function returns the distance between `str1` and `str2`.

You do NOT need to include pre-processor directives, function prototypes, or the main function. That is, you can start directly with the definition of the function.

```
#define MAX_STRING_LEN = 100;
#include <string.h>

...

int Dist(char *str1, char *str2)
{










}
```

Question 5 (7+7 marks)

Suppose that we have n persons, and we capture the friendships among them using a two-dimensional matrix F with n rows and n columns. In particular, if the i-th person and the j-th person are friends, then $F[i-1][j-1] = 1$; otherwise, $F[i-1][j-1] = 0$. We refer to F as the *friendship matrix*.

We say that two persons P_i and P_j are *connected*, if they are friends, or there exist a sequence of persons $P_1^*, P_2^*, \ldots, P_k^*$ such that (i) P_i and P_1^* are friends, (ii) P_j and P_k^* are friends, and (iii) for any $x \in [1, k-1]$, P_x^* and P_{x+1}^* are friends. Let C be a two-dimensional matrix with n rows and n columns, such that $C[i-1][j-1] = 1$ = 1 if the i-th person and the j-th person are connected, and $C[i-1][j-1] = 0$ otherwise. We refer to C as the *connection matrix*.

We need a function FtoC that generates the connection matrix C from the friendship matrix F. In particular, there are three input parameters to the function: an integer n, and two $n \times n$ matrices F and C. The function should examine F, and for each element in C, the function should decide whether it should be 1 or 0.

a) Write an algorithm in pseudocode for the function FtoC.

Question 5 (cont.)

b) Translate the algorithm in a) into a C function, using the following skeleton, i.e., FtoC. The function takes three input parameters: an integer n no more than 100, a two-dimensional matrix F where the first n rows and first n columns represent a friendship matrix, and an un-initialized two-dimensional matrix C. The function compute the connection matrix for the friendship matrix in F, and store the connection matrix in the first n rows and first n columns of C.

You do NOT need to include pre-processor directives, function prototypes, or the main function. That is, you can start directly with the definition of the function.

```
#define MAX_LEN 100

...

void FtoC(int n, int F[][MAX_LEN], int C[][MAX_LEN])
{
```

- END OF PAPER -